IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Serial No. 09/870,622
Confirmation No. 1783

SUPPLEMENTAL APPEAL BRIEF

Sir/Madam:

Further to the Notice of Appeal faxed August 24, 2004 and received in the U.S. Patent and Trademark Office on the same day, Appellant presents this Supplemental Appeal Brief. This supplemental brief is filed in response to a Notice of Non-Compliant Brief mailed October 13, 2006. The Notice of Appeal was filed following mailing of a Final Office Action on May 27, 2004. Appellant hereby appeals to the Board of Patent Appeals and Interferences from a final rejection of claims 1-17 in the Final Office Action, and respectfully requests that this appeal be considered by the Board.

I.   REAL PARTY IN INTEREST

The subject application is owned by International Business Machines Corporation, a corporation having its principal place of business at New Orchard Road, Armonk, New York, 10504, as evidenced by the assignment recorded at 011888, Frame 0535.

II.   RELATED APPEALS AND INTERFERENCES

A Notice of Appeal has been filed for the following application, which shares a common specification with the application currently on appeal.

09/870,615   Notice of Appeal filed 9/14/04

09/870,621   Notice of Appeal filed 9/24/04

However, because dissimilar art is cited in the present application and the above-mentioned related application, Appellants do not believe that the outcome of this appeal will have any bearing on the Board's decision on the related appeal. No other appeals or interferences are known which would directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

III.   STATUS OF CLAIMS

Claims 1-17 are pending in the captioned case. Claims 1-17 stand rejected.

IV.   STATUS OF AMENDMENTS

No amendments to the claims have been filed subsequent to their final rejection. The Appendix hereto therefore reflects the current state of the claims.

V.   SUMMARY OF CLAIMED SUBJECT MATTER

Appellant's claimed invention relates to a system of software components (82, 84 and 88, Fig. 8), a computer-readable storage device (18, Fig. 1) and a method (Figs. 8 and 10) for graphically displaying an object (AWT button object 30, Fig. 8), which was created by an application program (legacy APP 28) running under, and at least somewhat dependent on, an operating system (OS 40). In accordance with aspects of the present invention, however, the system of software components displays the object in a manner that is truly independent of the operating system. (Specification – page 24, line 8 to page 29, line 10, and Abstract).

More specifically, the presently invention overcomes the difficulties traditionally involved in migrating legacy applications from AWT to Swing by providing a functional extension of Swing – referred to herein as "AWTSwing" – which allows Swing components to be substituted for existing AWT components. The substitution of components enables the original AWT object to be redrawn or otherwise graphically displayed in a manner, which is independent from the operating system. (Specification – page 24, line 15 to line 29)

In one embodiment as recited in present claim 1, the system of software components employs a number of "lightweight software components" including, e.g., a graphics resource component (JButton class 48, Fig. 8), a proxy component (JButton Proxy 88) and a peer component (JButton Peer 82 and/or JComponent Peer 84). As noted in claim 1, the peer component is adapted for receiving events (e.g., key events, such as focus, mouse, sizing, etc.) pertaining to the object (AWT Button 30), and for routing the events to the proxy component. (see, Specification – page 25, lines 1-9, page 26, lines 1-3, page 28, line 18 to page 29, line to page 29, line 1 and Fig. 8). The proxy component associates the object with the graphics resource component and invokes methods of the graphics resource component for displaying the object in a manner, which is independent from the operating system. (see, Specification – page 25, line 23 to page 26, line 5, page 28, lines 3-5, page 29, lines 1-10 and Fig. 8). Because the lightweight graphics resource component does not use native code (i.e., OS-dependent code) for rendering images of the object upon a display, the graphics resource component is adapted to display the object independently of the operating system. (see, Specification – page 26, lines 1-5, page 28, lines 3-5, page 29, lines 1-10, and Fig. 8)

In some cases, the object (AWT Button 30, Figs. 8 and 10) may be part of a graphical user interface (see the GUI illustrated in Fig. 10). When the object is included within a particular layout (e.g., Frame 41, Figs. 8 and 10), the association of the object with the graphics resource component (JButton class 48, Fig. 8) establishes a parent-child relationship between the layout and the lightweight graphics resource component. (see, Specification – page 25, line 23 to page 26, line 1, Figs. 8 and 10, and claims 7 and 15). This parent-child relationship, which is established by the proxy component (JButton Proxy 88, Figs. 8 and 10), allows the graphics resource component to draw an image of a (lightweight) object over the image of the

(heavyweight) object, which was initially drawn with the aid of the windowing system of the operating system. (see, Specification – page 26, lines 1-5, page 29, lines 1-10, Figs. 8 and 10, and claims 8 and 16). In other words, the presently claimed system of software components allows images of lightweight (Swing) objects to be drawn or "painted" over images of heavyweight (AWT) objects.

In some embodiments, the system of software components may be stored within a computer-readable storage device. As noted in claim 17, the computer-readable storage device (18, Fig. 1) may include a windows-based operating system (OS 40, Figs. 1-2) and an application program (legacy APP 28) running under the operating system. (see, Specification – page 17, lines 17-20). In addition, the computer-readable storage device may include a graphics resource component (JButton class 48, Fig. 8), which is adapted to display an object (AWT Button 30) initially created by the application program. As described in more detail below, the graphical display of the object is performed independently of the windowing system of the operating system by invoking the method steps recited, e.g., in independent claim 9.

In some embodiments, the method as set forth in claim 9, may include the steps of: (i) utilizing a graphics resource component (JButton class 48, Fig. 8), which is adapted to display the object (AWT button 30, Fig. 8) independently of the windowing system of the operating system (OS 40, Figs. 1-2) (see, Specification – page 26, lines 1-5, page 28, lines 3-5, page 29, lines 1-10, and Fig. 8); (ii) creating a proxy component (JButton proxy 88, Fig. 8) and establishing an association between the object and the graphics resource component via the proxy component (see, specification – page 25, line 23 to page 26, line 5, page 28, lines 3-5, page 29, lines 1-10 and Fig. 8); (iii) receiving events (e.g., focus, mouse, sizing, etc.) pertaining to the object in a peer component (JButton peer 82, Fig. 8) and routing them to the proxy component (see, specification – page 25, lines 1-9, page 26, lines 1-3, page 28, line 18 to page 29, line to page 29, line 1 and Fig. 8); and (iv) in response to the events, invoking methods of the graphics resource component via the proxy component to display the object (see, specification – page 26, lines 1-5, page 28, lines 3-5, page 29, lines 1-10, and Fig. 8).

VI.   GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

1.   Claims 1-17 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over a web
        publication provided by Sun, entitled *Introducing Swing* (hereinafter "IS-SUN") in view
        of a web publication, entitled *Mixing Heavy and Light Components*, by Amy Fowler
        (hereinafter "M-SUN").

VII.   ARGUMENT

        The contentions of the Appellant with respect to the ground of rejection presented for
review, and the basis thereof, with citations of the statutes, regulations, authorities, and parts of
the record relied upon are presented herein for consideration by the Board. Details as to why the
rejections cannot be sustained are set forth below.

A.   Patentability of claims 1-6, 9-14 and 17 under 35 U.S.C § 103(a)

        Independent claims 1, 9 and 17, and claims dependent thereon, were rejected as being
unpatentable under 35 U.S.C. § 103(a) over a web publication provided by Sun, entitled
Introducing Swing (hereinafter "IS-SUN") in view of another web publication, entitled *Mixing
Heavy and Light Components*, by Amy Fowler (hereinafter "M-SUN").

        MPEP 2143 establishes the basic requirements in finding a *prima facie* case of
obviousness. As described, to establish a case of *prima facie* obviousness of a claimed
invention, three criteria must be met. First, there must be some suggestion or motivation, either
in the references themselves or in the knowledge generally available to one of ordinary skill in
the art to modify the references or to combine reference teachings. Second, there must be a
reasonable expectation of success. Finally, the prior art references must teach or suggest <u>all</u> the
claim limitations. Specifically, "all words in a claim must be considered when judging the
patentability of that claim against the prior art." *In re Wilson* 424 F.2d 1382, 1385 (CCPA
1970). If an independent claim is nonobvious under 35 U.S.C. 103, then any claim depending
therefrom is nonobvious. *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988). Using

these standards, Applicants contend that the cited art fails to provide teaching, suggestion or motivation for all features of the currently pending claims, and furthermore, cannot be modified to do so. Several distinctive features of the present invention are set forth in more detail below.

IS-SUN and M-SUN each fail to provide teaching or suggestion for a system (claim 1), computer-readable storage device (claim 17), or method (claim 9) for graphical display of an object, where the system includes a peer component, which is adapted to receive events pertaining to the object and route the events to a proxy component, which associates the object with a graphics resource component and invokes methods thereof to display the object in a manner independent from an operating system. Independent claims 1, 9, and 17 each recite limitations on a graphics resource component, a peer component and a proxy component. These components are included within a new system of software components, referred to as "AWTSwing." As described in more detail below, IS-SUN and M-SUN each fail to provide teaching or suggestion for the presently claimed peer and proxy components.

The article entitled *Introducing Swing* does just as the title implies – it briefly outlines the Swing architecture and the basic differences between Swing and AWT. However, *Introducing Swing* (otherwise referred to as IS-SUN) does not provide teaching or suggestion for a system, computer-readable storage device, or method for graphical display of an object, where the system comprises a peer component, which is adapted to receive events pertaining to the object and route the events to a proxy component, which associates the object with a graphic resource component and invokes methods thereof to display the object in a manner independent from an operating system, as taught in present claims 1, 9, and 17.

Statements in the Office Action mailed November 21, 2003 and in the final Office Action mailed May 27, 2004 admit that IS-SUN fails to provide teaching or suggestion for the presently claimed peer and proxy components. For example, the Examiner clearly states, "IS-SUN … doesn't teach a proxy component which associates the object with the graphics resource component and invokes methods of the graphics resource component to display the object, or a peer component, adapted to receive events pertaining to the object and route the events to the proxy." (*See*, Office Action, page 2; Final Office Action, pages 2-3). Appellants sincerely

appreciate the Examiner's recognition of the lack of teaching within IS-SUN for the peer and proxy components recited in present claims 1, 9, and 17.

Though the Examiner admits to the lack of teaching within IS-SUN for the presently claimed peer and proxy components, the Examiner suggests that such teaching can be found within the article entitled *Mixing heavy and light components* (otherwise referred to as M-SUN). For example, the Examiner states, "M-SUN teaches a method of implementing swing components similar to that of IS-SUN, but further teaches a proxy component that associates an object with a graphics resource component, and further displays the object, in that the proxy component is the swing class (see page 2, paragraph 2), and a peer component, adapted to receive events pertaining to the object and route the events to the proxy component, in that the peer component is the ancestor (see page 2, paragraph 2)." (See, Office Action, page 2; final Office Action, page 3). In addition, the Examiner suggests that "it would have been obvious to one of ordinary skill in the art… to modify the swing component interface of IS-SUN to include the combinational properties as did M-SUN." (Office Action, page 3, Final Office Action, page 3). The Appellants respectfully disagree with the Examiner's interpretation of peer and proxy components, and further assert that IS-SUN and M-SUN cannot be modified or combined, for at least the reasons set forth in more detail below.

The article entitled *Mixing heavy and light components* (M-SUN) discloses various problems that arise when programmers attempt to mix heavyweight AWT components with lightweight Swing components. However, and as described in more detail below, M-SUN fails to provide any teaching, suggestion or motivation for the presently claimed peer and proxy components. Therefore, M-SUN <u>cannot be combined</u> with IS-SUN to overcome the deficiencies therein.

Some of the problems disclosed by M-SUN are similar to those disclosed in the present specification. For example, M-SUN mentions some of the difficulties encountered when one attempts to add lightweight Swing components to an AWT layout by "painting" or "drawing" images of the Swing components over pre-existing AWT images. On page 3, M-SUN states that "[w]hen a lightweight component overlaps a heavyweight component, the heavyweight

component is always on top, regardless of the relative z-order of the two components." On page 4, M-SUN illustrates how images rendered by heavyweight AWT components will obscure any overlapping images rendered by lightweight Swing components, and explicitly states that a "Swing label will never appear to be on top of [an] AWT label because the Swing label is rendering its contents in the native window of the frame (its first heavyweight ancestor), while the AWT label is rendering its contents in its own native window, which is actually a child of the frame's native window." As such, M-SUN recognizes a fundamental limitation of Swing, which does not permit images of Swing components to be drawn over AWT renderings.

The presently claimed case provides a solution to the above-mentioned problem by creating a proxy component, which is not included within the existing classes of software components currently belonging to the Swing application program interface (API). The presently claimed proxy component enables Swing components to be successfully incorporated within AWT-based application programs by associating existing heavyweight (AWT) objects with lightweight (Swing) graphics resource components. This enables the proxy component to invoke the methods of the lightweight graphics resource components for displaying the objects independently from the operating system. The presently claimed peer component differs from typical Swing peer components by routing events intended for the existing heavyweight object to the presently claimed proxy component. *See*, Specification, page 24, line 8 - page 29, line 10.

Unlike the presently claimed case, M-SUN provides no solution to the above-mentioned problem other than sheer avoidance. For example, M-SUN states, "[b]ecause there's sometimes no alternative to mixing heavyweight and lightweight components, we have provided a few options in Swing to make a certain level of component-mixing possible." (M-SUN, page 3, paragraph 2). These options, or guidelines, include: "[d]o not mix lightweight (Swing) and heavyweight (AWT) components within a container where the lightweight component is expected to overlap the heavyweight one." (M-SUN, page 5). From the above guideline, it is obvious that M-SUN chooses to avoid problematic situations, rather than present actual solutions to the problem. Since M-SUN fails to provide any solution of his own, M-SUN cannot be relied upon to disclose the presently claimed solution. In other words, M-SUN simply fails to provide

any teaching, suggestion or motivation for the peer and proxy components recited in present claims 1, 9, and 17.

In light of the Examiner's suggestion that "the proxy component [of M-SUN] is the swing class" and "the peer component [of M-SUN] is the ancestor," the Appellants wish to clarify the terms "class" and "ancestor" as they are used in the context of object-oriented programming and the presently claimed case. As described on pages 4 and 5 of the specification, a "class" is a fundamental entity in an object-oriented programming language that possesses certain attributes (e.g., shape, color, size, location on screen, behavior, etc.), which are common to all "objects" belonging to the same class. The objects can be developed as modular building blocks, with subsequent objects referred to as "children" of higher-level "parent" objects. In some cases, a parent object may include a "class" of child objects, and the child objects may inherit the class attributes (i.e., the properties and methods) of the parent object, or "ancestor", from which they derive. In the embodiment of Fig. 8, for example, Component 32 may be considered a "class" of objects, where one of the objects in the class is Button 30. Component 32 may also be considered a "parent" or "ancestor" of Button 30, since various methods and/or properties of Button 30 may be derived from Component 32.

However, a "Swing class" cannot be considered equivalent to a "proxy component" as suggested by the Examiner, since a "Swing class" merely <u>defines the properties and methods</u> (e.g., shape, color, size, location on screen, behavior, etc.) of a collection of Swing objects, whereas a "proxy component" actually <u>functions to associate an object</u> (e.g., a button displayed in a GUI) with a graphics resource component (e.g., JButton 48) and to <u>invoke the methods of the graphics resource component</u> (e.g., run the program code contained within JButton 48 for <u>displaying the object</u>. A "Swing class," in itself, does not and cannot function to associate an object with a graphics resource component, invoke the methods of the graphics resource component, or display the object. Therefore, the mere mention of Swing classes within M-SUN does not provide teaching or suggestion for the proxy component recited in present claims 1, 9, and 17.

Furthermore, though a "peer component" may be an "ancestor" of some other object, merely stating so provides no evidence of the peer component being adapted to receive events pertaining to the object and to route the events to a proxy component. Since M-SUN fails to provide teaching for a "proxy component", any peer components that may be described by M-SUN cannot be configured to route events to a non-existent proxy component. As such, M-SUN also fails to provide teaching or suggestion for the peer component recited in present claims 1, 9, and 17.

The above arguments were presented in response to the Office Action Mailed November 21, 2003. After reviewing the above arguments, the Examiner maintained his position in the final Office Action by providing yet another interpretation of the term "proxy component." For example, statements in the final Office Action now suggest that "the swing class [of M-SUN] can be considered a proxy component in the sense that a proxy can mean an element that acts as a substitute for another element (swing for AWT)." (Final Office Action, page 7). As described in more detail below, the Appellants disagree with the Examiner's latest interpretation of a "proxy component," and further assert that a "Swing class" cannot be considered a "proxy component," as it is defined by the presently claimed case.

The Appellants recognize that, "[d]uring examination, the claims must be interpreted as broadly as their terms reasonably allow.'" (See, MPEP 2111.01). However, the Appellants assert that the pending claims cannot be interpreted more broadly than their terms reasonably allow, or be given a meaning that is inconsistent with the specification. The specification and present claims 1, 9 and 17 each describe the presently claimed "proxy component" as one that associates an object with a graphics resource component and invokes methods of the graphics resource component to display the object. Even when given its plain meaning, it is simply unreasonable for one skilled in the art to interpret the term "proxy component" as "an element that acts as a substitute for another element" since neither the specification nor the claims support such an interpretation. As such, the Examiner's interpretation of the term "proxy component" is considered to be improper and erroneous.

In addition, the mere mention of Swing classes within M-SUN cannot be relied upon to provide teaching or suggestion for a proxy component, regardless of the manner in which the term "proxy component" is interpreted.   As noted above, a "Swing class" merely defines the common properties and methods (e.g., shape, color, size, location on screen, behavior, etc.) associated with a collection of Swing objects that descend from the Swing class. However, a "Swing class" does <u>not</u> function to associate an object with a graphics resource component, nor does a "Swing class" invoke methods of the graphics resource component for displaying the object. Therefore, the mere mention of Swing classes with M-SUN does not provide evidentiary support for the presently claimed "proxy component", <u>as it is defined by the specification and the present claims</u>. Furthermore, and contrary to the Examiner's suggestions in the final Office Action, Swing classes are <u>not</u> "elements that act as substitutes for other elements," and more specifically, do not substitute Swing components for AWT components. Therefore, a "Swing class" cannot be considered equivalent to a "proxy component," <u>even as it is improperly defined by the Examiner</u>.

Further statements in the final Office Action appear to suggest that the presently claimed proxy component may be inherently taught by M-SUN. For example, the Examiner suggests that since M-SUN "teaches a combination of two APIs … in order to combine two APIs there must be a connecting element." (Final Office Action, page 7). Again, Appellants assert that it is improper to oversimplify and loosely interpret the presently claimed proxy component as a "connecting element," since such interpretation is significantly more broad than allowed by the language used in claims 1, 9, and 17. Therefore, Appellants maintain the assertion that absolutely no teaching or suggestion for the presently claimed proxy component can be found, either explicitly or inherently described, within the sole reference (M-SUN) which the Examiner relies upon to provide such teaching.

There is no motivation to combine or modify the teachings of IS-SUN and M-SUN to provide the presently claimed peer and proxy components. As explained above, IS-SUN and M-SUN each fail to provide teaching or suggestion for the presently claimed peer and proxy components, and therefore, <u>cannot be combined</u> to do so. Even if the cited art were improperly

combined, the combined cited art would still fail to provide teaching or suggestion for the peer and proxy components described in present claims 1, 9, and 17.

In addition, the cited art <u>cannot be modified</u> to teach or suggest the aforementioned limitation, since IS-SUN and M-SUN each fail to suggest a desirability for doing so. The mere fact that references can be combined or modified does not render the resultant combination obvious unless the prior art also suggests the desirability of the combination [or modification]. *In re Mills*, 916 F.2d 680, 16 USPQ2d 1430 (Fed. Cir. 1990); MPEP 2143.01.

Unlike the presently claimed case, IS-SUN fails to mention the difficulties involved in mixing heavyweight AWT components and lightweight Swing components. Consequently, IS-SUN provides no desirability for overcoming such difficulties by providing lightweight peer and proxy components, as described in claims 1, 9 and 17. Though M-SUN does admit to some difficulties in mixing heavyweight AWT components and lightweight Swing components, M-SUN chooses to deal with the difficulties by <u>avoiding the problematic situations altogether</u>. By choosing avoidance, M-SUN fails to provide motivation for a system or method that actually solves the problem. Since IS-SUN and M-SUN each fail to provide motivation to teach or suggest the aforementioned limitations of the presently claimed case, IS-SUN and M-SUN cannot be modified to do so.

Obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so. *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed.Cir. 1988); *In re Jones*, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992); MPEP 2143.01. Appellants assert that the teaching or suggestion to make the claimed combination [or modification] must be found in the prior art, and not based on Applicant's own disclosure. *In re Vaeck*, 947 F.2d 488, 20 USPQ2d 1438 (Fed. Cir. 1991); MPEP 2142.

B.   <u>Patentability of claims 7, 8, 15, and 16 under 35 U.S.C § 103(a)</u>

Claims 7-8 and 15-16 were rejected under 35 U.S.C. § 103(a) as being unpatentable over IS-SUN in view of M-SUN. Because claims 7-8 and 15-16 are dependent from independent claims 1 and 9, respectively, the arguments presented above for patentability of claims 1 and 9 apply equally to claims 7-8 and 15-16, and are herein incorporated by reference. In addition to the § 103 arguments presented above with respect to claims 1 and 9, additional arguments are provided below to establish patentability of dependent claims 7-8 and 15-16.

None of the cited art teaches or suggests that, when the object is part of a layout, the association of the object with the graphics resource component can be used to establish a parent-child relationship between the layout and the graphics resource component, which allows the graphics resource component to draw a (lightweight) object over an existing image of the (heavyweight) object originally drawn with the aid of the windowing system of the operating system. Claims 7-8 and 15-16 place a further limitation on the presently claimed proxy component introduced in claims 1 and 9. For example, claims 7 and 15 state that when an object is part of a layout, the association of the object with the graphics resource component establishes a parent-child relationship between the layout and the graphics resource component. As set forth in claims 8 and 16, the parent-child relationship (established by the proxy component) allows lightweight (i.e., Swing) graphics resource components to "draw" or "paint" over existing images of heavyweight (i.e., AWT) objects, which were originally drawn with the graphical resources of the operating system.

As noted above, IS-SUN and M-SUN each fail to disclose the presently claimed peer and proxy components, and furthermore, cannot be modified or combined to do so. As described in more detail below, IS-SUN and M-SUN also fail to provide teaching or suggestion for the additional limitations placed on the presently claimed proxy component, as set forth in dependent claims 7, 8, 15, and 16.

Statements in the final Office Action rely on M-SUN for teaching the limitations of dependent claims 7, 8, 15, and 16. For example, the final Office Action states, "M-SUN teaches, in page 4, paragraph 2 and page 6, paragraph 2 and the following picture, that if heavyweight components are used it is possible for them to obscure what is drawn by the windowing system of the operating system." (Final Office Action, page 5). The Appellant agrees with the Examiner's current interpretation of M-SUN. Heavyweight (AWT) components included within prior art legacy applications will, in fact, obscure any lightweight (Swing) components, which are purposefully or inadvertently placed underneath the heavyweight components. For example, M-SUN illustrates the case in which a heavyweight label is drawn on top of a lightweight label on page 4.

However, the Examiner's current interpretation is completely inconsistent with the limitations recited in dependent claims 7, 8, 15, and 16, which describe how the parent-child relationship established by the proxy component allows <u>lightweight</u> graphics resource component (*see*, claims 1 and 9) <u>to draw over an existing image of a (heavyweight) object drawn with the aid of the operating system</u>. Because M-SUN clearly states that lightweight Swing objects "will never appear to be on top" of heavyweight AWT objects (*see*, pages 3-4 of M-SUN), M-SUN cannot be relied upon to provide teaching or suggestion for the additional limitations set forth in dependent claims 7, 8, 15, and 16.

There is no motivation to modify or combine the cited art to teach or suggest the presently claimed parent-child relationship. As noted above, none of the cited art provides teaching or suggestion for the presently claimed parent-child relationship, which allows a (lightweight) graphics resource component to draw over an existing image of a (heavyweight) object drawn with the aid of the operating system, as described in claims 7-8 and 15-16.

In addition to the lack of teaching, Appellants assert that the cited art cannot be combined or modified to teach or suggest the presently claimed parent-child relationship, since the cited art fails to suggest a desirability for doing so. The mere fact that references can be combined or modified does not render the resultant combination obvious unless the prior art also suggests the

desirability of the combination. *In re Mills*, 916 F.2d 680, 16 USPQ2d 1430 (Fed. Cir. 1990); MPEP 2143.01.

As noted above, M-SUN fails to provide a manner in which lightweight graphics resource components can be used to successfully draw over images previously drawn with the aid of the operating system. Instead, M-SUN strongly recommends that programmers "[d]o not mix lightweight (Swing) and heavyweight (AWT) components within a container where the lightweight component is expected to overlay the heavyweight one." (M-SUN, page 4, paragraph 3). By choosing avoidance, rather than presenting a solution to the problem, M-SUN provides no absolutely desirability for the presently claimed parent-child relationship. As such, M-SUN cannot be modified to teach or suggest the additional limitations set forth in dependent claims 7, 8, 15, and 16.

For at least the foregoing reasons, Appellant asserts that independent claims 1, 9, and 17, as well as claims dependent therefrom, are patentably distinct over IS-SUN and M-SUN. Contrary to the characterizations made in the various Office Actions, the cited references cannot be properly combined or modified to provide teaching for all limitations recited in claims 1, 9, and 17. Accordingly, Appellants assert that a *prima facie* case of obviousness has not been duly set out and, therefore, request that this rejection be reversed.

* * *

For the foregoing reasons, it is submitted that the Examiner's rejection of claims 1-17 was erroneous, and reversal of the decision is respectfully requested.

The Commissioner is authorized to charge the required fees (if any) to Daffer McDaniel LLP deposit account number 50-3268.

Respectfully submitted,
/Kevin L. Daffer/
Kevin L. Daffer
Reg. No. 34,146
Attorney for Appellant

Customer No. 35617
Date: November 7, 2006
JMF

VIII. CLAIMS APPENDIX

The present claims on appeal are as follows.

1.   A system for graphical display of an object created by an application program running
under an operating system, comprising:

        a graphics resource component adapted to display the object independently of the
                operating system;

        a proxy component, which associates the object with the graphics resource component
                and invokes methods of the graphics resource component to display the object;
                and

        a peer component, adapted to receive events pertaining to the object and route the events
                to the proxy component.

2.   The system as recited in claim 1, wherein the peer component is independent of the
operating system, and emulates the behavior of a second peer component that employs the
windowing system of the operating system.

3.   The system as recited in claim 1, wherein the object is part of a graphical user interface
(GUI) associated with the application program.

4.   The system as recited in claim 3, wherein a look and feel of the GUI is independent of the
operating system.

5.   The system as recited in claim 1, wherein the application program is written in Java
programming language.

6.   The system as recited in claim 1, wherein the proxy extends an existing class of software components belonging to the Swing application program interface (API).

7.   The system as recited in claim 1, wherein the object is part of a layout, and the association of the object with the graphics resource component establishes a parent-child relationship between the layout and the graphics resource component.

8.   The system as recited in claim 7, wherein the parent-child relationship between the layout containing the object and the graphics resource component allows the graphics resource component to draw over an existing image of the object drawn with the aid of the windowing system of the operating system.

9.   A method for graphical display of an object created by an application program running under an operating system, comprising:

    utilizing a graphics resource component adapted to display the object independently of the windowing system of the operating system;

    creating a proxy component and establishing an association between the object and the graphics resource component via the proxy component;

    receiving events pertaining to the object in a peer component and routing them to the proxy component; and

    in response to the events, invoking methods of the graphics resource component via the proxy component to display the object.

10.   The method as recited in claim 9, wherein the peer component is independent of the operating system, and the method further comprises emulating the behavior of a second peer component that employs the windowing system of the operating system.

11.   The method as recited in claim 9, wherein the object is part of a graphical user interface (GUI) associated with the application program.

12.   The method as recited in claim 11, wherein a look and feel of the GUI is independent of the operating system.

13.   The method as recited in claim 9, wherein the application program is written in Java programming language.

14.   The method as recited in claim 9, wherein the proxy extends an existing class of software components belonging to the Swing API.

15.   The method as recited in claim 9, wherein the object is part of a layout, and the method further comprises using the association of the object with the graphics resource component to establish a parent-child relationship between the layout and the graphics resource component.

16.   The method as recited in claim 15, further comprising using the parent-child relationship between the layout containing the object and the graphics resource component to allow the graphics resource component to draw over an existing image of the object drawn with the aid of the windowing system of the operating system.

17.   A computer-readable storage device, comprising:

        a windows-based operating system;

        an application program running under the operating system;

        a graphics resource component adapted to display an object created by the application
                program independently of the windowing system of the operating system, by:

creating a proxy component and establishing an association between the object and the graphics resource component via the proxy component;

receiving events pertaining to the object in a peer component and routing them to the proxy component; and

in response to the events, invoking methods of the graphics resource component via the proxy component to display the object.

IX. EVIDENCE APPENDIX

No evidence has been entered during the prosecution of the captioned case.

X. RELATED PROCEEDINGS APPENDIX

A Notice of Appeal has been filed for the following application, which shares a common specification with the application currently on appeal.

09/870,615   Notice of Appeal filed 9/14/04

09/870,621   Notice of Appeal filed 9/24/04

However, because dissimilar art is cited in the present application and the above-mentioned related application, Appellants do not believe that the outcome of this appeal will have any bearing on the Board's decision on the related appeal. No other appeals or interferences are known which would directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.